Cozmo Song Match PROGRAMMERS GUIDE

Brought to you by Cozmo Design Studios

Casey Boyer, Shaon Islam, Sam Schuler, Sefik Mehmedovic, and Dishant Shrestha

INDEX

INTRODUCTION

Game Objective

Song Match is a game where you try to match the notes of a song by tapping blocks with Cozmo. There are three separate game modes: Song Match (the main game), Tutorial Mode, and Ear Training Mode.

Currently, since Cozmo only supports three cubes, songs with only three notes are used.

Links

The source code for the expanded Song Match game, with the new features added by Cozmo Design Studios, can be found here

The User guide for the expanded Song Match game can be found here:

The source code for the original Song Match game, implemented by the Cozmonauts, can be found here

INITIAL SETUP

To use the Cozmo SDK, the Cozmo mobile app must be installed on your mobile device and that device must be tethered to a computer via USB cable.

Prerequisites

- Python 3.5.1 or later
- Wi-Fi connection
- An iOS or Android mobile device with the Cozmo app installed, connected to the computer via USB cable

SDK Example Programs

Anki provides example programs for novice and advanced users to run with the SDK. Download the SDK example programs here:

- macOS/Linux SDK Examples
- Windows SDK Examples

Once downloaded, extract the packaged files to a new directory.

Installation

To install the SDK on your system, see the instructions for your computer's operating system.

- Installation - macOS / OS X
  http://cozmosdk.anki.com/docs/install-macos.html
- Installation – Windows
  http://cozmosdk.anki.com/docs/install-windows.html
- Installation – Linux
  http://cozmosdk.anki.com/docs/install-linux.html#install-linux

TROUBLESHOOTING

Failure to Install Python Package

If your attempt to install Python packages such as NumPy fails, please upgrade your pip install as follows:

On macOS and Linux:

```
pip3 install -U pip
```

On Windows:

```
py -3 -m pip install -U pip
```

Once the pip command is upgraded, retry your Python package installation.

Cozmo SDK Forums

Please visit the Cozmo SDK Forums to ask questions, find solutions and for general discussion.

INSTALLATION MACOS / OS X

This guide provides instructions on installing the Cozmo SDK for computers running with a macOS operating system.

Installation Videos

For your convenience, videos are provided showing the installation steps being followed on a macOS / OS X computer; one using an iOS device, and one using an Android device. There is also full text-based documentation below these.

https://youtu.be/zNgmUwuHq-M

https://youtu.be/6xWuhOtkfsc

Python Installation

1. Install Homebrew on your system according to the latest instructions. If you already had brew installed, then update it by opening a Terminal window and typing in the following:
2. brew update
3. Once Homebrew is installed and updated, type the following into the Terminal window to install the latest version of Python 3:
4. brew install python3

SDK Installation

To install the SDK, type the following into the Terminal window:

```
pip3 install --user 'cozmo[camera]'
```

Note that the `[camera]` option adds support for processing images from Cozmo's camera.

INSTALLATION MACOS / OS X *CONTINUED*

SDK Upgrade

To upgrade the SDK from a previous install, enter this command:

```
pip3 install --user --upgrade cozmo
```

Mobile Device Setup

- **iOS** devices do not require any special setup in order to run the Cozmo SDK on a macOS system.
- **Android** devices require installation of Android Debug Bridge (adb) in order to run the Cozmo SDK. This is required for the computer to communicate with the Android mobile device over a USB cable and runs automatically when required.

Troubleshooting

Please see the Troubleshooting section of the Initial Setup page for tips or visit the Cozmo SDK Forums to ask questions, find solutions, or for general discussion.

INSTALLATION WINDOWS

This guide provides instructions on installing the Cozmo SDK for computers running with a Windows operating system.

Installation Videos

For your convenience, videos are provided showing the installation steps being followed on a Windows computer; one using an iOS device, and one using an Android device. There is also full text-based documentation below these.

https://youtu.be/gtRS3iqzSuA

https://youtu.be/9TJeK_AEFYo

Python Installation

Download the Python 3.5.1 (or later) executable file from Python.org and run it on your computer.

Important

We recommend that you tick the “Add Python 3.5 to PATH” checkbox on the Setup screen.

SDK Installation

To install the SDK, type the following into the Command Prompt window:

```
pip3 install --user cozmo[camera]
```

Note that the `[camera]` option adds support for processing images from Cozmo’s camera.

INSTALLATION WINDOWS *CONTINUED*

SDK Upgrade

To upgrade the SDK from a previous install, enter this command:

```
pip3 install --user --upgrade cozmo
```

Mobile Device Setup

- **iOS** devices require iTunes to ensure that the usbmuxd service is installed on your computer. Usbmuxd is required for the computer to communicate with the iOS device over a USB cable. While iTunes needs to be installed, it does not need to be running.
- **Android** devices require installation of Android Debug Bridge (adb) in order to run the Cozmo SDK. This is required for the computer to communicate with the Android mobile device over a USB cable and runs automatically when required.

Troubleshooting

Please see the Troubleshooting section of the Initial Setup page for tips or visit the Cozmo SDK Forums to ask questions, find solutions, or for general discussion.

LINUX INSTALLATION

This guide provides instructions on installing the Cozmo SDK for computers running with an Ubuntu Linux operating system.

Warning

The Cozmo SDK is tested and supported on Ubuntu 14.04 and 16.04. Anki makes no guarantee the Cozmo SDK will work on other versions of Linux. If you wish to try the Cozmo SDK on versions of Linux *other than* Ubuntu 14.04 or 16.04, please ensure the following dependencies are installed:

- Python 3.5.1 or later
- pip for Python 3 (Python package installer)
- Android command-line tools (https://developer.android.com/studio/index.html#Other)
- usbmuxd for iOS / Android Debug Bridge for Android

Ubuntu 14.04

Python Installation

1. Type the following into your Terminal window to install Python 3.5:

```
sudo add-apt-repository ppa:deadsnakes
sudo apt-get update
sudo apt-get install python3.5 python3.5-tk
```

2. Then, install and set up `virtualenv`:

```
sudo apt-get install python-pip
pip install virtualenv
virtualenv -p python3.5 ~/cozmo-env
```

3. Last, make sure to activate the `virtualenv` any time you use cozmo:

```
source ~/cozmo-env/bin/activate
```

4. Note: You'll need to activate the `virtualenv` before running any python or pip commands. Learn more about `virtualenv` here.

LINUX INSTALLATION

CONTINUED

Ubuntu 16.04

Python Installation

1. Type the following into your Terminal window to install Python:

```
sudo apt-get update
sudo apt-get install python3
```

2. Then install pip by typing in the following into the Terminal window:

```
sudo apt install python3-pip
```

3. Last, install Tkinter:

```
sudo apt-get install python3-pil.imagetk
```

SDK Installation

To install the SDK, type the following into the Terminal window:

```
pip install 'cozmo[camera]'
```

Note that the `[camera]` option adds support for processing images from Cozmo's camera.

SDK Upgrade

To upgrade the SDK from a previous install, enter this command:

`pip install --upgrade cozmo` (For Ubuntu 14.01)

`pip3 install --user --upgrade cozmo` (For Ubuntu 16.04)

LINUX INSTALLATION

CONTINUED

Mobile Device Setup

- **iOS** devices require usbmuxd in order to run the Cozmo SDK. Usbmuxd is required for the computer to communicate with the iOS device over a USB cable.
- **Android** devices require installation of Android Debug Bridge (adb) in order to run the Cozmo SDK. This is required for the computer to communicate with the Android mobile device over a USB cable and runs automatically when required.

Troubleshooting

Please see the Troubleshooting section of the Initial Setup page for tips or visit the Cozmo SDK Forums to ask questions, find solutions, or for general discussion.

ANDROID DEBUG BRIDGE

Important

Android Debug Bridge (adb) is required ONLY for those users pairing an Android mobile device with Cozmo. Android Debug Bridge (adb) is necessary for the Cozmo SDK to work with Android mobile devices. Please ensure adb is installed before attempting to run the Cozmo SDK with a paired Android device.

macOS Installation

1. Type the following into a Terminal window (requires Homebrew to be installed):

```
brew tap caskroom/cask
brew cask install android-platform-tools
```

2. Continue to Final Installation (All Platforms) below to complete installation.

ANDROID DEBUG BRIDGE *CONTINUED*

Windows Installation

1. In your internet browser, navigate to this link and download file `platform-tools-latest-windows.zip` to your Downloads folder.
2. Open a File Explorer window to your Downloads folder and see that your downloaded file `platform-tools-latest-windows.zip` is there.
3. Open a new File Explorer window and create a new folder in `C:\Users\your_name` named `Android`. Then, navigate into your new Android folder. You should now be inside folder `C:\Users\your_name\Android`.
4. Move the zip file you downloaded in step 1, `platform-tools-latest-windows.zip`, to your new Android folder at `C:\Users\your_name\Android`.
5. Right-click the `platform-tools-latest-windows.zip` file in `C:\Users\your_name\Android` and select Extract All.
6. With File Explorer, navigate to `C:\Users\your_name\Android\platform-tools-latest-windows\platform-tools` and confirm that the adb is there. You will now add this path to your PATH environment variable in the next step.
7. Add adb to your PATH environment variable.
8. Right-click the Start menu and select *System*.
9. Select *Advanced System Settings -> Advanced -> Environment Variables*.
10. Under *User Variables*, select *PATH* and click **Edit**.
11. Under *Edit Environment Variables*, click **New** and add the path to the folder containing adb (e.g. `C:\Users\your_name\Android\platform-tools-latest-windows\platform-tools`). Click OK on all dialog boxes to confirm your change.
12. Confirm that the PATH is correctly pointing to adb.
13. Open new a Command Prompt window. (To find the Command Prompt, you may use the search box in the lower left-hand corner of your screen.)
14. Type `adb` and adb instructions should print out.
15. Continue to Final Installation (All Platforms) below to complete installation.

ANDROID DEBUG BRIDGE *CONTINUED*

Linux Installation

1. In your internet browser, navigate to this link and download file `platform-tools-latest-linux.zip`. You may need to accept the license before downloading.
2. Navigate to the zip file download location (e.g., ~/Downloads) and extract the files. The files will be extracted to folder `platform-tools`.
3. In your home directory, create folder `android-sdk-linux`:
   ```
   cd ~
   mkdir android-sdk-linux
   ```
4. Move the extracted `platform-tools` folder to your new `android-sdk-linux` folder:
   ```
   mv Downloads/platform-tools android-sdk-linux
   ```
5. Confirm you see adb inside `platform-tools`:
   ```
   cd android-sdk-linux/platform-tools
   ls adb
   ```
6. Add adb to your PATH.
7. Edit your *~/.bashrc* file and add this line:
   ```
   export PATH=${PATH}:~/android-sdk-linux/platform-tools
   ```
8. Save *.bashrc* and then call:
   ```
   source .bashrc
   ```
9. Confirm that adb is in your PATH by calling the following command:
   ```
   which adb
   ```
10. The result of this command should be the path where adb is installed.
11. Continue to Final Installation (All Platforms) below to complete installation.

FINAL INSTALLATION (ALL PLATFORMS)

1. Enable USB Debugging on your phone.
    a. On Android devices:
        i. Tap seven (7) times on the Build Number listed under *Settings -> About Phone*.
        ii. Then, under *Settings -> Developer Options*, enable USB debugging.
    b. On Amazon Kindle Fire:
        i. Tap seven (7) times on the Serial Number listed under *Settings -> Device Options*.
        ii. Then, under *Settings -> Device Options -> Developer Options*, turn on Enable ADB.
2. Connect your Android device to your computer via USB. When the *Allow USB Debugging?* popup displays, tap **OK**.
3. At the command line, type this command to confirm that your device shows:

    adb devices

At least one device should show in the result, for example:

List of devices attached

88148a08 device

If you are required to accept the connection request on the mobile device itself, a message will appear saying the device is unauthorized. For example:

List of devices attached

88148a08 unauthorized

GETTING STARTED WITH THE COZMO SDK

To make sure you get the best experience possible out of the SDK, please ensure you have followed the steps in the Initial Setup.

Cozmo SDK Forums

Please visit our Cozmo SDK Forums where you can:

- Get assistance with your code
- Hear about upcoming SDK additions
- Share your work
- Join discussion about the SDK
- Be a part of the Cozmo SDK Community!

Starting Up the SDK

1. Plug the mobile device containing the Cozmo app into your computer.
2. Open the Cozmo app on the phone. Make sure Cozmo is on and connected to the app via Wi-Fi.
3. Tap on the gear icon at the top right corner to open the Settings menu.
4. Swipe left to show the Cozmo SDK option and tap the Enable SDK button.
5. Make sure the SDK examples are downloaded from the Downloads page.
6. On the computer, open Terminal (macOS/Linux) or Command Prompt (Windows) and type `cd cozmo_sdk_examples`, where `cozmo_sdk_examples` is the directory you extracted the examples into, and press Enter.

GETTING STARTED WITH THE COZMO SDK

CONTINUED

Setup

1. Setup the Cozmo SDK.
    a. Please see Initial Setup under the Cozmo SDK documentation.
2. Clone the repository.
    a. $ git clone https://github.com/samuelschuler/CS4500.git
3. Navigate to the repository.
    a. $ cd cozmo-song-match
4. Install dependencies.
    a. $ pip install -r requirements.txt
5. Download and Install PyCharm (Optional)

GETTING STARTED WITH THE COZMO SDK

CONTINUED

Starting the Game

With Cozmo connected, run the main program. For details on connecting Cozmo, see Getting Started With the Cozmo SDK.

```
$ python main.py
```

We support command line arguments for configuring what song you want to play, and how many players. Use the help flag `-h` for help.

```
$ python main.py -h

usage: main.py [-h] [-s S] [-p N]

Play Song Match with Cozmo.

optional arguments:

-h, --help: show this help message and exit

-s S: The song to play. Hot Cross Buns (hcb), Mary Had A Little Lamb (mhall), Rain Rain Go Away (rrga), (ir) It’s Raining It’s Pouring, or (ra) Ring Around the Rosie. Defaults to a random song.

-p N: The number of players for the game, where N is the number of players. Defaults to None. If None then selecting the number of players will be handled in game.
```

NEW FEATURES

Multiple Game Modes

Song Match Mode

With each round, the game gets longer and a little harder. Song Match supports up to 3 players.

The game starts by playing the first three notes of a song. Each time a note is played, a corresponding cube flashes. The player must match the notes by tapping the correct sequence of cubes. If the player gets the sequence correct, then Cozmo tries to match the correct sequence. If either the player or Cozmo gets *three* notes incorrect, then they lose the game.

This makes up **one** round. Each round the length of the sequence increases, until you reach the end of the song.

Tutorial Mode

The tutorial mode is a mode where one player will go through an entire song step by step with Cozmo to understand how the game works. Cozmo will point at the cube to be tapped. In this mode, the light of this cube will flash white five times and will stay white until tapped. If the player taps the incorrect cube(s) three times, Cozmo will show them the correct sequence. It is a collaborative game.

Ear Training Mode

The ear-training mode creates a song with three randomly chosen notes. Then, song is created by randomly creating a sequence from these three chosen notes. The sequence of notes, which is the entire song, may be anywhere between 5-10 notes long. Additionally, each note in the song sequence is randomly chosen to be a whole note, half note, quarter note, or an eighth note. The goal of the game is to train players to audibly recognize notes.

NEW FEATURES *CONTINUED*

Additional Instruments

The original Song Match game only supported one instrument, Piano. We implemented an `instrument` class to support an unlimited number of instruments. Currently, the Song Match game has the following instruments: Piano, Clarinet, and Flute. Now, an instrument is randomly selected when a song is played, rather than defaulting to Piano.

To add more instruments, create a new package (named according to your instrument) under the `sound_effects` package, and add the instrument .wav files to this package. Then, in the `instrument` class under the `song_match.song` package, declare your instrument names as strings, and add them to the `instrument_dictionary` member of the class.

Warning: when adding new instruments, make sure that the added instrument(s) have the notes required to play any of the predefined songs. For example, if an instrument is added that does not have the note 'D4', then the instrument cannot play the song Ring Around the Rosie, since this is a note in the song. As a result, we added a unit testing for each new instrument added, which you can find in the `test_note_instrument` class under the `test` package. This should clarify the differences between the instruments, and which instruments support what notes.

Cube Pagination Menu

Previously, the only time Cozmo asked for the user input in the form of cube taps was when asking how many players and during the gameplay. Now, we have modified the `option_prompter` class under the `song_match` package such that it:

- accepts a list of three `str` elements representing options that Cozmo will prompt the user with
- accepts a list of three `str` elements representing the Cozmo's response based on the option selected
- the return value will represent the cube tapped, and is used to obtain the option selected by the user

In doing so, we were able to implement a pagination for selecting the game mode and for whether the user wishes to continue playing after a game is over.

NEW FEATURES *CONTINUED*

New Songs

We added two more songs, **Ring Around the Rosie** and **It’s Raining It’s Pouring**. Now, the Song Match game has 5 total songs to choose from:

- Mary Had a Little Lamb
- Hot Cross Buns
- Rain Rain Go Away
- Ring Around the Rosie
- It’s Raining It’s Pouring

FUTURE EXPANSIONS

More Instruments

Currently, only one instrument plays a given song. While any of the songs may play with any of the three instruments, it would be interesting for one song to play with multiple instruments. This may involve assigning a unique instrument to each of the cubes, assigning an instrument for specific notes in the song sequence, etc. With this implementation, the player(s) would also have to pay closer attention to the instruments in addition to the notes being played.

Multiple Notes per Cube

Cozmo will only recognize the first distinct three cubes placed in front of him. This means he will not recognize additional cubes. So, adding more cubes will not allow songs to be created with more than three notes. Thus, it would open a wider range of songs to implement if each cube could be associated with multiple notes rather than one note per cube. This would allow you to create songs with 4, 5…,*n* notes. Currently, the three note limitation is a severe hinderance, as not many recognizable three note nursery rhymes exist. Naturally, this leads to implementing difficulty modes such as easy, medium, and difficult, depending on the note complexity of a song.

Ear Training Mode Expansion

The Ear Training Mode is quite similar to the Song Match Mode, with the main difference being that the song is randomly created. To better train the player(s) to recognize different note pitches, this mode could be restructured to have the user guess certain pitches played back in a sequence. Additionally, this mode could be modified to have Cozmo prompt the user to play a specific note in the sequence, so that the user must focus more on the sound of the notes rather than the order they are played in.

FUTURE EXPANSIONS

CONTINUED

Modify Sequence Playback

In both the Song Match Mode and Ear Training Mode, the game does not end until either Cozmo or the player(s) repeat the sequence incorrectly three times; this means that if the player(s) and Cozmo repeat the sequence correctly for the entire song, the game will not end until the entire song is played. Although the maximum number of notes in a song sequence is not more than 20, this can take a long time to play through – and is difficult for age groups above the 4-6 age range. So, the sequence playback should be modified in such a way that the user can more easily repeat the song sequence.

PYCHARM

The following aims to explain why we recommend using PyCharm to develop Song Match.

If you haven't already, download and install PyCharm Community Edition here:

https://www.jetbrains.com/pycharm/download/

Finding Usages

The first reason we recommend you use PyCharm is the ability to find usages. Often when developing you need to find all the places where a function is called. PyCharm makes this easy.

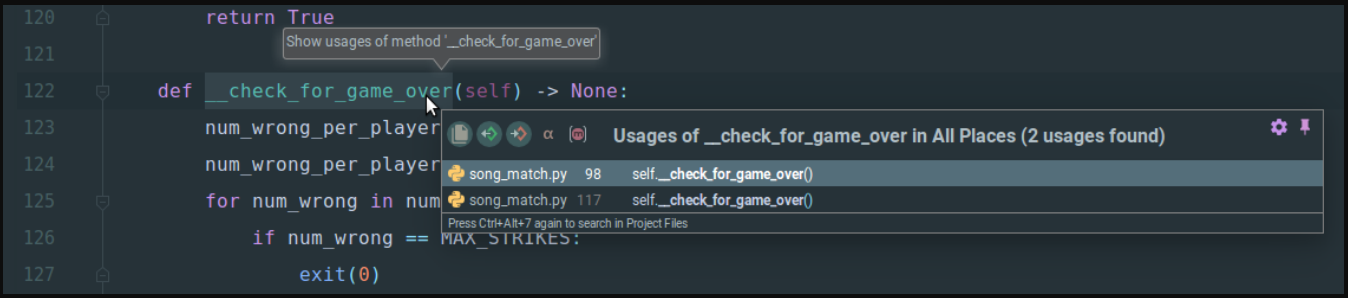
For example, here's a function that checks for whether the game is over in `song_match.py`:

```
120         return True
121
122     def __check_for_game_over(self) -> None:
123         num_wrong_per_player = [player.num_wrong for player in self._players]
124         num_wrong_per_player.append(self._song_robot.num_wrong)
125         for num_wrong in num_wrong_per_player:
126             if num_wrong == MAX_STRIKES:
127                 exit(0)
```

If you hover over the name of the function while holding `Ctrl`, and click the name of the function you can find usages.

PyCharm also has other keyboard shortcuts and ways to find usages that are helpful to learn.

PYCHARM CONTINUED



The tooltip window shows the function is called in two places:

- Line 98 in `song_match.py`
- and line 117 in `song_match.py`

You can select either usage and PyCharm will automatically navigate there for you.

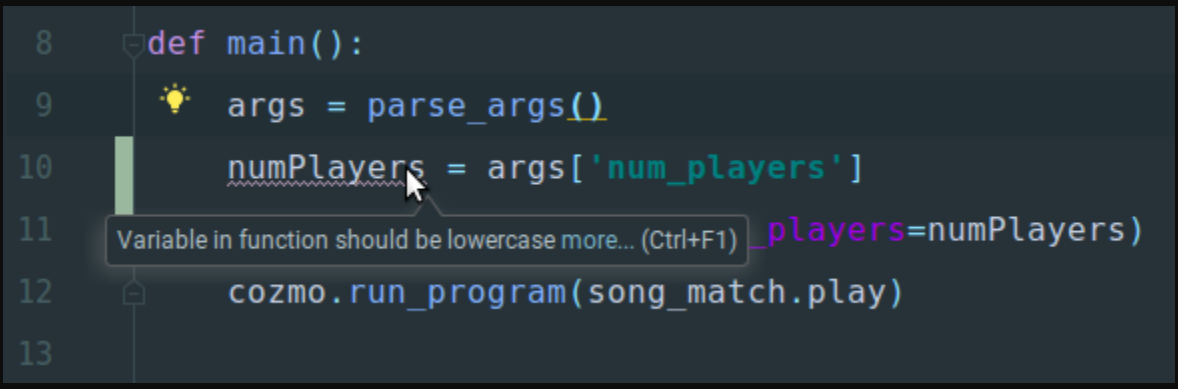
Note

Finding usages works for functions, methods, variables, and anything else you could care about.

Enforcing PEP 8

PyCharm also helps developers follow Python's official style guide PEP 8.

For example, PEP 8 states variables should use snake case instead of camel case.



PyCharm underlines the camelcase variable with a yellow squiggly line and suggests renaming it from `numPlayers` to `num_players`.

PYCHARM CONTINUED

So Much More

PyCharm offers many other great features that are outside the scope of this document like:

- Integration with `requirement.txt` files
- Powerful refactoring abilities
- Creating and Optimizing imports
- Reformatting source code
- and so much more.

CODEBASE CONVENTIONS

The following aims to explain some of the common conventions found within the Song Match codebase.

Design Patterns

The Song Match codebase is almost entirely object oriented. There are many design patterns relating to object-oriented programming (OOP) found throughout the codebase.

Wrapper Objects

The first design pattern you'll see are our use of wrapper objects. A "wrapper object" is an object that takes an already instantiated object in its constructor and extends its functionality through custom methods.

More formally this is known as the Decorator Pattern because your decorating an object by wrapping it and adding behavior.

Common objects from the Cozmo SDK, like the objects representing Cozmo and the cubes, have a corresponding wrapper object in the Song Match codebase:

- `SongRobot` wraps `Robot`
- `NoteCube` wraps `LightCube`
- `NoteCubes` wrap a list of 3 `LightCube` instances

Many methods and properties of the wrapper objects match the corresponding wrapped object. For example, both `NoteCube` and `LightCube` have a `set_lights` method. The `set_lights()` method in `NoteCube` simply calls `set_lights()` on the internal `LightCube` object.

```
# note_cube.py

def set_lights(self, light: Light):

    self._cube.set_lights(light)
```

OBJECT CREATION PATTERNS

Static Factory Methods

A static factory method is a static method used for creating an object.

There are two static factory methods in the codebase. Both are methods named `of`, a concise naming convention for static factory methods popularized by `EnumSet` in Java. See How to name factory like methods? for more details.

- `NoteCube` – `of()`
- `NoteCubes` – `of()`

Factories

A factory is an object for creating other objects.

In our codebase there is one factory, `EffectFactory`.

`EffectFactory` creates our various game effect subclasses:

- `CorrectSequenceEffect` - Played when a player matches the correct notes.
- `RoundTransitionEffect` - Played when transitioning between game rounds.
- `WrongNoteEffect` - Played when a player fails to match the correct notes.

Inheritance

We favor composition over inheritance and avoid complex class hierarchies.

No class extends an instantiable class, but there are two abstract base classes:

- `Effect` - Abstract base class for various game effects
- `Song` - Abstract base class for various songs

OBJECT CREATION PATTERNS *CONTINUED*

Public, Protected, and Private

Python lacks access modifiers like `private` and `protected` found in languages like Java and C#.

We follow the convention of preceding `private` methods and attributes with two underscores. For example:

```
def __some_private_method():
    pass
```

`protected` methods and attributes are preceded with a single underscore.

```
def _some_protected_method():
    pass
```

If you see anything that begins with an underscore, then it means don't use it outside of that class or module.

In general, all `public` members in Song Match have docstring comments, while `private` and `protected` members do not.

Type Hinting

In general, all functions and methods are type hinted. Below we see a function that adds two `int` values together, and returns an `int`.

```
def add_two_numbers(a: int, b: int) -> int:
    return a + b
```

See support for type hints for more details.